FAQ

NFC/RFID R E A D E R M O D U L E @ 13,56 MHz

Frequently Asked Questions 0.5

ARYGON TECHNOLOGIES

Erthalstrasse 1
D - 55118 Mainz
Germany
Phone +49 (0) 61 31-30476-0
Fax +49 (0) 61 31-30 476-20
info@arygon.com • http://www.arygon.com

Document: Frequently Asked Questions

Author: J. Schneider / T. Baur / A. Kretschmann

Current Version: 0.5

Date: April 2008

Document history

Revision	Date	Description
0.1	September 2006	First published version
0.2	October 2006	I2C behaviour was added
0.3	February 2007	§ 3.2.2 e) LL-command `MxRtyPassiveActivation` are described
0.4	August 2007	Merged with document “Abstract 0.1”
0.5	April 2008	FAQ n) and o) added. RS232 interface a) changed.

Contact information

For additional information and sales office addresses visit:

http://www.arygon.com or **http://www.nfc-global.com**

Contents

1. Description

About this document

This document summarises some questions which we have got from our customers. Get in contact with us if you have any further questions or ideas in order to improve our support.

2. Conventions and notations

2.1 Abbreviations

ACMA	ARYGON Core Module Version A
APPA	ARYGON Plug and Play Module Version A
RM	Reader Module
SMX	SmartMX
p2p	peer to peer (passive, active)
NFC	Near Field Communication
PN531	Philips Reader-IC
TAMA	Philips Reader-IC, nickname for PN53x
NA	Not applicable
PICC	Mifare® proximity card
API	Application Programming Interface
IDE	Integrated Development Environment
VCP	Virtual com port

2.2 Document information

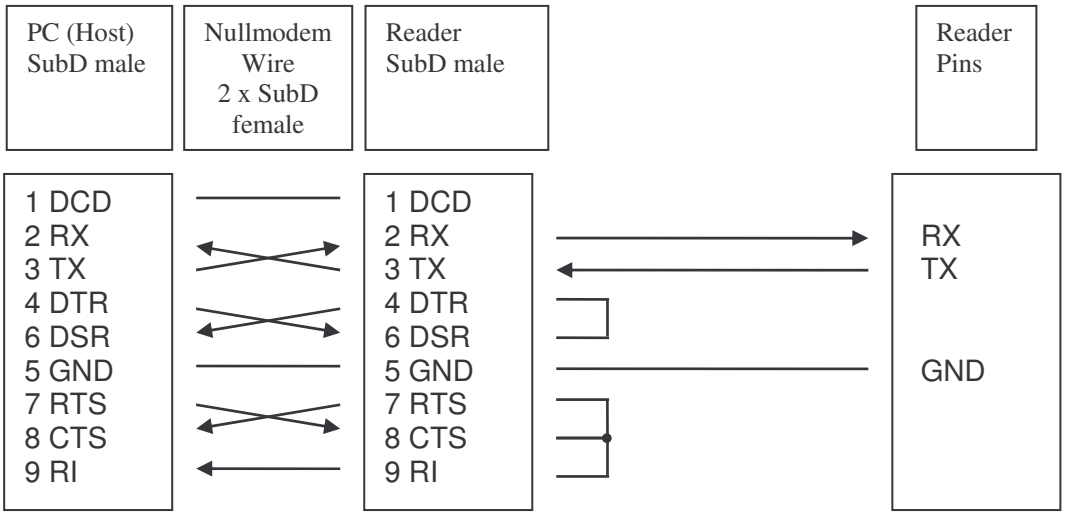
Info	Content
Keywords	MIFARE®, NFC, PN531
Abstract	This document describes the Frequently Asked Questions.

3. Frequently Asked Questions
3.1 Hardware

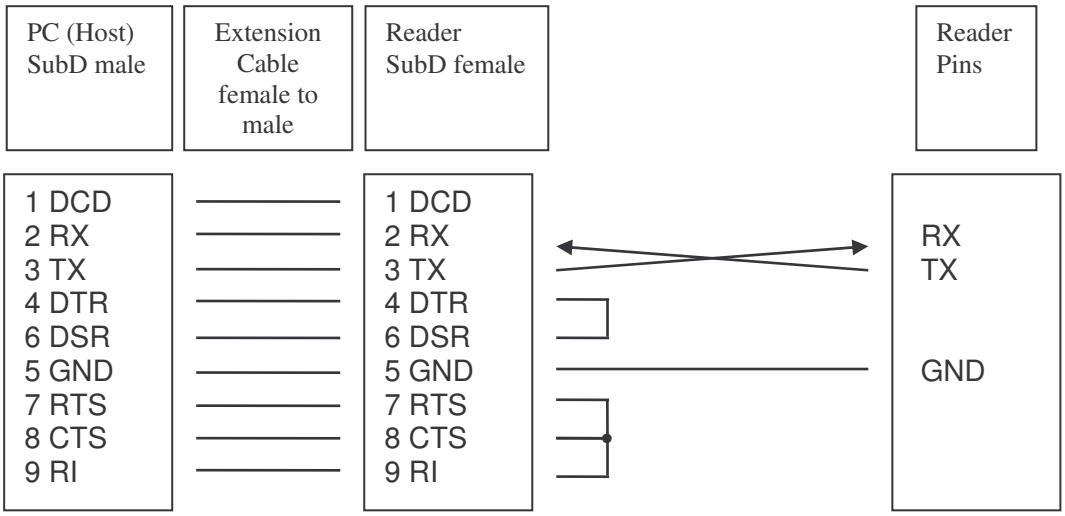
a) What do I have to consider when connecting an ARYGON reader via RS232 interface (e.g. APPA23205)?

Some PC hardware and software "dislike" floating levels on their open RS232 Com Port pins. To avoid problems the null modem principle is strongly recommended. Otherwise many unnecessary interrupts (level CTS, DSR or RI) can occur. If a user wants to connect the RS232 reader direct to a standard PC, the following wiring should be considered.

Nullmodem principle with Nullmodem cable:



Nullmodem principle with Straight Through cable:

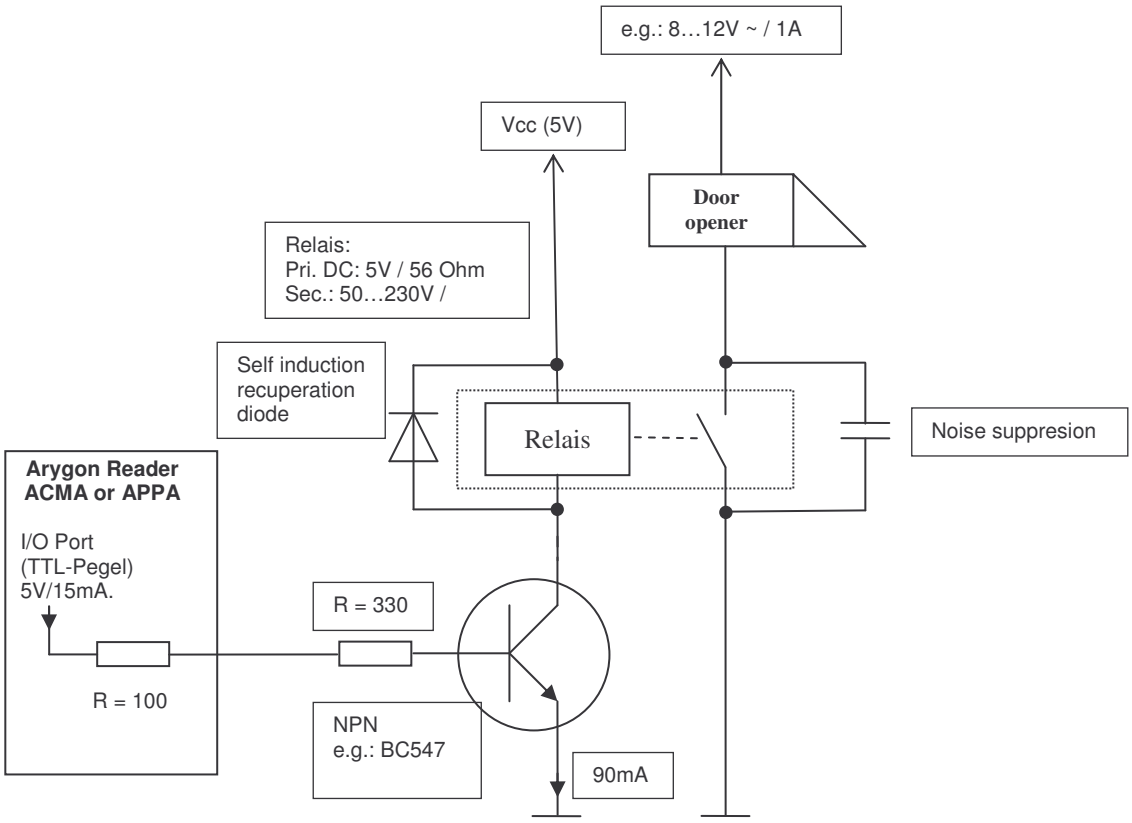


b) Is it possible to use the ARYGON NFC/Mifare reader module as door opener?

Yes, of course, but you have to add an additional circuit.

The user I/O ports supply only a current of max. 15mA. Therefore an additional amplification is necessary to get a minimum current of 1A.

Example of an I/O port amplifier schematic



c) Which kind of LCD display supports the ARYGON reader module and how can I use it?

As hardware interface we use 9 pins to the Samsung KS0066 or HD44780 display controllers or compatible.

- 4-bit bi-directional data bus.
- 3 control lines.
- 2 pins for power supply (+5V and GND).

All those pins are provided by the ARYGON reader module. It supports 2 x 20 or 2 x 40 characters. The command sets are in ASCII command as well.

Get in touch with us to get the document ANSW_Display_10.pdf

3.2 Software

a) Which protocol modes are available as host interface.

ARYGON distinguishes between two languages:
- ARYGON high level language
- TAMA low level language

The availability of languages depends on the mounting of the printed circuit board. Therefore you have to make a difference between “µC equipped” and “no µC equipped”.

The availability of the µC is *mandatory* in case of:

- Stand-alone applications like access control.
- Wiegand™ interface.
- Party line applications (RS485 / RS422 / I2C).
- High level language usage in any terminal program (for easier testing).
- High level language usage in Mifare® applications where checksum handling is not required or not possible because of host limitations.
- Personalization of the Reader. Only with µC equipped it is possible to store the Mifare® authentication keys into the Reader (µC EEPROM). TAMA provides no user EEPROM.
- Storing the NFC Open Identifier data.
- DES / 3DES encryption/decryption for Desfire Tags or for NFC peer to peer communications with ciphering.
- More than two user GPIO pins needed for extra digital switching or digital read functions. (TAMA offers two user GPIOs, µC offers 9 user GPIOs, a PWM output pins and an analog input pin.)
- Standard LCD interface needed.
- Performing a TAMA hardware reset via a software command from host to µC.

Protocol modes (µC equipped)

Several protocol modes are supported via a **mode select command byte** which is sent as first Byte in every data packet from host to the µC. The mode select command byte tells the equipped µC how the following bytes have to be interpreted.
The mode select command byte is always sent in **ASCII format**.
Following mode select command Bytes are available:

'**0**' High level language in ASCII format.
(Common µC commands and Mifare® commands)

'**1**' High level language in Binary format with addressing byte for party line.
(Common µC commands and Mifare® commands)

'**2**' Philips protocol (TAMA language) in binary format.

'**3**' Philips protocol (TAMA language) in binary with addressing byte for party line.

b) Can you show me the ARYGON high level language command set ?

Commands for adjustments and special µC-functions are always started with letter 'a'.

Common µC commands (auxiliary commands)	
'ah'	Set UART baud rate to host side.
'at'	Set UART baud rate to TAMA side.
'ar'	Initiate a TAMA hardware reset.
'au'	Initiate a µC software reset.
'av'	Get µC firmware version.
'asn'	Get the unique serial number of the reader.
'as'	TAMA RFRegulation test.
'asl'	Power down mode (sleep).
'apc'	Configuration of the GPIO pins.
'apw'	Write GPIO.
'apr'	Read GPIO.
'apm'	Set PWM output duty cycle.
'apa'	Read analog port input value.
'ali'	Internal µC EEPROM login to modify EEPROM data.
'alo'	Internal µC EEPROM logout.
'aer'	Read one Byte from the internal µC EEPROM.
'aew'	Write one Byte into the internal µC EEPROM.
'aek'	Write a Mifare® login keytype with key into the internal µC EEPROM.
'aec'	Check appropriate EEPROM Mifare® key location if it is used.
'aep'	Write a new PIN code for internal µC EEPROM write access.
'aen'	Write a new PIN code for internal µC EEPROM NFC open identifier area.
'aes'	Write (set) one block of data into the NFC open identifier area.
'aeg'	Read (get) one block of data from the NFC open identifier area.
'apl'	Party line (RS4XX) polling command.

ISO 14443-A / Mifare commands	
's'	Select a single card/tag
'l'	Login (authenticate)
'r'	Read data block/page on a tag
'wb'	Write data block/page (16 Byte)
'w4'	Write data block/page (4 Byte) for Mifare® ultralight tags
'rv'	Read value block
'wv'	Write/format value block
'+'	Increment value block
'-'	Decrement value block
'='	Copy value block
'h'	Set tag into halt
'of'	RF Configuration (switch off antenna)
'p'	Polling for tags with autonomous periodic wake up from sleep mode. (= Select a single card with reduced power consumption).

c) What does the response packet mean in case of high-level command 'select a single card/tag'?

PC -> RM

```
0 s          //0 = mode 0; s=select a single card/tag)
```

RM -> PC

```
FF0000 16 4B 01 01 04 00 08 04 32EEED2E
```

Case: Mifare® standard 1k:

FF0000: Status, refer to µC response packet.
16: User data length (number of character in Hex).
4B: TAMA InListPassiveTarget Response packet.
01: Number of initialized targets.
01: Target Number
0400: SENS_RES = Answer to request
08: SEL_RES (card Type) (08 = Mifare® standard) = answer to select
04: Card ID (NFCID1) length.
32EEED2E: Card ID (NFCID1). (Refer to PN531 User Manual)

Case: Mifare® Ultralight

FF0000: Status, refer to µC response packet.
1E: User data length (number of character in Hex).
4B: TAMA InListPassiveTarget response packet.
01: Number of initialized targets.
01: Target number
4400: SENS_RES
00: SEL_RES (card Typ) (00 = Mifare® UltraLight)
08: Card ID (NFCID1) length.
8804686211127A00: Card ID (NFCID1).

Case: Mifare® standard 4k

FF0000: Status, refer to µC response packet.
16: User data length (number of character in Hex).
4B: TAMA InListPassiveTarget response packet.
01: Number of initialized targets.
01: Target number
0200: SENS_RES
18: SEL_RES (card Typ) (18 = Mifare® 4k)
04: Card ID (NFCID1) length.
56347400: Card ID (NFCID1).

d) How can I switch on/off the LED on the ARYGON desktop reader ADRA?

Our LEDs are no power LED. They are controlled by the user application with the ARYGON high-level language.

LED commands:

Set the I/O LED Ports to output (only once):

```
0 a p c 0 2 0 0     // red LED
0 a p c 0 6 0 0     // green LED
```

Switch LED to on:

```
0 a p w 0 2 0 1      // red LED
0 a p w 0 6 0 1      // green LED
```

Switch LED to off:

```
0 a p w 0 2 0 0      // red LED
0 a p w 0 6 0 0      // green LED
```

Refer to ANSW-ACMA-APPA-xx.pdf for more info about the reader command set.

e) Is there a DLL or API for the ARYGON reader modules available?

No. The reader protocol can be considered as an “open interface” protocol. You have the possibility to write your own application without reservations or limitations. An API or DLL is not needed as you can communicate with the reader modules by using the Com port support of your IDE.

f) Which drivers are available for the ARYGON desktop reader or USB Readers?

The USB interface of ARYGON reader modules can be accessed via the virtual com port (VCP) or direct driver support (USBExpress) of the following operating systems.

- Windows 98 SE/2000/XP/Vista
- Linux 2.4 or later (Please read the txt-file ..\SOFTWARE\SiLabs_CP210xDriverInstall\LINUX\NOTE_LINUX_DRIVER.txt
- Macintosh OS
- WindowsCE

More Information of our used USB to UART converter IC:

http://www.silabs.com

g) Can I read DESFire cards with the ARYGON reader modules?

Yes, a new firmware version with integrated ARYGON High Level DESFire command set including 3DES decryption is now available (Beta Version).
Ask for our new DESFire application Note: ANSW-DESFire-xx.pdf.

h) I set my NFC enabled phone (e.g. Nokia or Samsung) to Mifare 1k (virtual card mode), but the ARYGON NFC/Mifare reader doesn't respond the expected answer. What's wrong?

The reader sends automatically the first command of ISO14443 when the card indicates it supports it. Consequently, after "select a single card/tag" command, the NFC enabled phone is in ISO14443-4 mode, and doesn't understand anymore Mifare command.

The problem is just on the reader side, you must send the TAMA low-level command SetTAMAParameters.

Indeed, the NFC enabled phone is able to communicate either in mifare mode or in ISO14443-4 mode. (bit 5 of the SEL_RES = 0x28 means ISO14443-4 support, as defined in ISO14443-4 standard specification).

So, on the reader side, if you want to communicate in mifare mode, you have to disable the automatic selection of ISO14443-4:

```
// low level language
PC -> RM: SET TAMA PARAMETERS
      00 00 FF 03 FD D4 12 04 16 00
RM -> PC: ACK
      00 00 FF 00 FF 00
RM -> PC: SET TAMA PARAMETERS EXECUTED
      00 00 FF 02 FE D5 13 18 00
```

Default mode (ISO14443-4):

```
PC -> RM: INITIATOR:
0s                              // µC Select a single Card 106 kBaud Card
RM -> PC
FF 00 00 00                     // Response - OK!
RM -> PC
FF 00 00 30 4B 01 01 04 00 28 04 30 44 1C 80 0D 78 80 80 02 00 73 C8 40 00
00 90 00                        // Response NFC enabled Phone
```

Mifare mode:

```
// low level language
PC -> RM: SET TAMA PARAMETERS
      00 00 FF 03 FD D4 12 04 16 00
RM -> PC: ACK
      00 00 FF 00 FF 00
RM -> PC: SET TAMA PARAMETERS EXECUTED
00 00 FF 02 FE D5 13 18 00

// High Level in ASCII
PC -> RM: INITIATOR:
      0s                        // µC Select a single Card 106 kBaud card
RM -> PC
      FF 00 00 00               // Response - OK!
RM -> PC
      FF 00 00 16 4B 01 01 04 00 28 04 F0 E1 1C 80
                                // Response of the NFC enabled phone
```

i) Which Mifare commands are available in case of TAMA low-level language?

Mifare command set	PCD (TAMA)	Command set supported by TAMA
PICC_REQIDL	0x26	TAMA only intern
PICC_REQALL	0x52	TAMA only intern
PICC_ANTICOLL1	0x93	TAMA only intern
PICC_ANTICOLL11	0x92	TAMA only intern
PICC_ANTICOLL12	0x94	TAMA only intern
PICC_ANTICOLL13	0x98	TAMA only intern
PICC_ANTICOLL2	0x95	TAMA only intern
PICC_ANTICOLL3	0x97	TAMA only intern

->The command 4A 01 00 (InListPassiveTarget) masks the commands mentioned before.

PICC_AUTHENT1A	0x60	40 01 60 15 FF FF FF FF FF FF 82 63 58 32
PICC_AUTHENT1B	0x61	40 01 61 20 FF FF FF FF FF BB F2 07 EE 2E
PICC_READ16	0x30	40 01 30 15
PICC_WRITE16	0xA0	40 01 A0 20 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0F 10 11
PICC_WRITE4	0xA2	40 01 A2 21 01 02 03 04
PICC_DECREMENT	0xC0	40 01 C0 21 04 00 00 00;
PICC_INCREMENT	0xC1	40 01 C1 21 04 00 00 00; //
PICC_RESTORE	0xC2	40 01 C2 15;
PICC_TRANSFER	0xB0	40 01 B0 16;
PICC_HALT	0x50	44 01; TAMA InDeselect command
RF off	32 01 00;	

(Refer to PN531 User Manual)

In case of Increment, Decrement or Restore (copy), there is a combination with command TRANSFER necessary.

j) Is there a source code for developing with the mounting of "no uC equipped" available?

Yes, following you find our source code for building the TAMA command frames.

Building of frame structure from host (µC) to TAMA (refer to PN531 User Manual):

```
// info begin
typedef struct
   {
   uchar ptr;   // read/write pointer
   uchar buff[BUFLENGTH2];
   } rxstruct;

typedef struct
   {
   uchar anz_data; // 0 = empty, buflength = full
   uchar rd_ptr;   // read pointer
   uchar wr_ptr;   // write pointer
   uchar buff[BUFLENGTH];
   } rngbuffstruct;  // for rng_wr_txbuf1
// info end


void build_tama_frame(rxstruct *buff)
   {
   uchar checks, anz, i;
   /* folgend: uchar Typecast => autom. int erweiterung verhindern */
   /* EasyCODE - */
   anz = buff->ptr + 1;
   rng_wr_txbuf1(0x00);          // preamble
   rng_wr_txbuf1(0x00);          // start code
   rng_wr_txbuf1(0xFF);          // start code
   rng_wr_txbuf1(anz);           // LEN
   rng_wr_txbuf1(((uchar)-anz)); // LEN checksum
   rng_wr_txbuf1(0xD4);          // TFI (from Host to TAMA)
   checks = 0xD4;
   for (i = 0; i < (buff->ptr); i++)
      {
      rng_wr_txbuf1(buff->buff[i]);
      checks += buff->buff[i];
      }
   rng_wr_txbuf1(((uchar)-checks)); // packet data checksum
   rng_wr_txbuf1(0x00); // postamble
   TX1IE = 1;
   }
```

TAMA frame check, after receiving a TAMA response:

```
uchar  tama_checks(rxstruct *buff)
/*
Reponse:

TRUE:
Checksum - ok!

FALSE:
Checksum - wrong (packet length or packet data)
*/
   {
   uchar checks ,i;


   /* following: uchar Typecast => autom. int enhancement are prohibited */
   if (buff->buff[3] != 0)
      {
      if (buff->buff[4] != ((uchar)-buff->buff[3]))
         {
         // packet length checks wrong
         return FALSE;
         }
      checks = 0;
      for (i = 0; i < buff->buff[3]; i++)
         {
         checks += buff->buff[i+5];
         }
      if (( buff->buff[buff->buff[3] + 5] ) != ((uchar)-checks))
         {
         // packet data checks wrong
         return FALSE;
         }
      if (buff->buff[5] == 0x7F)
         {
         error1 = 9;
         }
      }
   else
      {
      // Tama ack packet
      }
   return TRUE;
   }
```

k) What do I have to consider in case of connecting an ARYGON Reader via I²C (e.g. APPA1I205) ?

At the beginning it should be mentioned that TAMA is not comparable with an I2C EEPROM!

1.)
As first, according the I2C specification accidentally bit changes because of EMC influence are not allowed.
The complete circuit must ensure that all lines are kept very short (max. 10cm) and that they are absolutely stabled and protected. I2C is not comparable with UART or RS232 / RS485 or USB because line buses are tolerant for bit interferences I2C absolutely not !
In I2C environment all devices can hang if for example the ACK bit, Start or stop condition will fail.

2.)
We have observed that sometimes the TAMA I2C ACK Bit is missing but this is allowed!

If the polling read command is sent to fast to the TAMA and the TAMA is busy, then it happens that the TAMA sends no ACK bit.

According to I2C specification:
"When a slave doesn't acknowledge the slave address (for example, it's unable to receive or transmit because it's performing some real-time function), the data line must be left HIGH by the slave. The master can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer."

In order to understand the behaviour, please read the following investigation.

We use a PC I2C Test tool, we are sending to the TAMA the command, get Firmware Version:
`00 00 FF 02 FE D4 02 2A 00`

With an oscilloscope we have traced the following pins:
- **Yellow: TAMA SCL**.
- **Green: TAMA SDA**.
- **Magenta**: Test point on your interface board = direction of SDA

low means: TAMA can send or host wants to sent high
high means: host wants to send low)

```
48
00 00 FF 02 FE D4 02 2A 00
49  (TAMA I2C ACK Bit is missing here)
```

Delay of about 13ms.

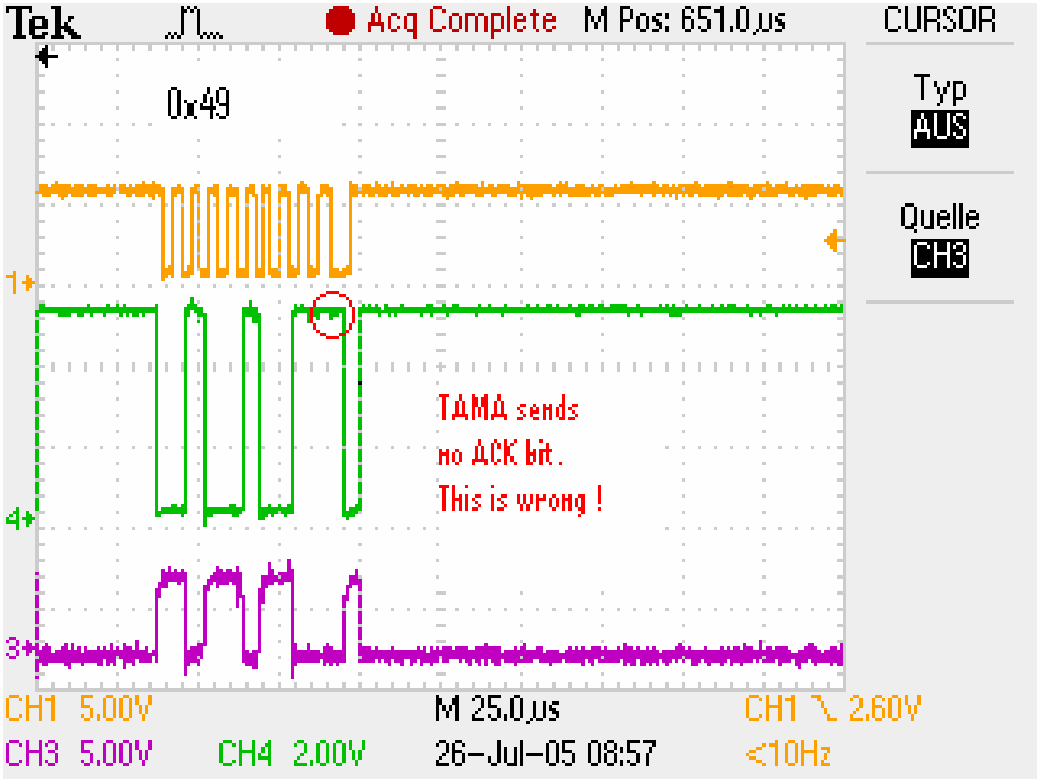
```
49
01
```
TAMA ACK packet
TAMA response packet

It seems all is **OK**.

After every Data Byte the TAMA sends the I2C ACK bit (low) to the PC – It's **OK**.

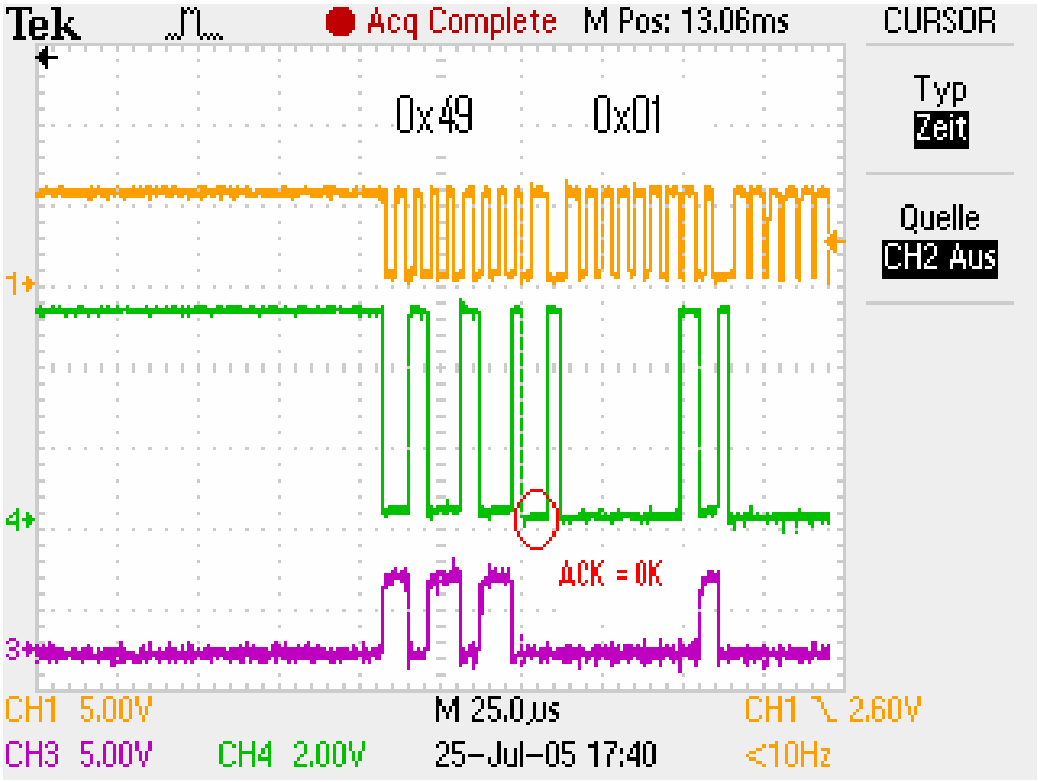
0x49 is the *Read* command is **OK** (first polling)
But the TAMA does not set the I2C ACK bit (low) after receiving of the first **0x49**.
Then the PC Tool does not activate the CLOCK (SCL), then the TAMA doesn't send the status Byte with RDY = 0x00

The ACK bit is missing:



Because of the fact that the PC Tool does ignore this problem, sometimes the packet flow continues.

Then about 10...20ms later the PC Tool activates the SCL (clock).
And now we see a correct I2C ACK bit (low) from the TAMA.
And now we see the correct Status Byte 0x01 = RDY from the TAMA.



And the packet flow is continued.

The above behaviour is no bug, it is according to I2C specification.

3).
Sometimes it happens that the TAMA will pull down the SCL line and The I2C host is blocked (hangs).

This is allowed we found out and according I2C specification.

According to I2C specification:
"If a Slave device cannot handle incoming data until it has performed some other function, it can hold SCL low to force the Master into a wait-state."

Conclusion:
If you consider all these facts I2C works. For testing we have programmed our µC with I2C interface for TAMA I2C testing and we have no problem any more.

l) What can I do with the low-level command `MxRtyPassiveActivation`?

If a function is necessary to get directly a response"***Card detected – YES / NO***", in this case you can send a low-level command `MxRtyPassiveActivation :`

Step 1
PC -> RM: `MxRtyPassiveActivation`

```
00 00 FF 06 FA D4 32 05 FF 00 02 F4 00          // 0x02 = 3 times

MxRtyPassiveActivation = 0x00 is not possible.
```

Step 2
PC -> RM: INITIATOR:

```
      0s                        // µC Select a single Card 106 kBaud card
```

RM -> PC

```
      FF 00 00 00               // Response - OK!
```

Card detected:
RM -> PC

```
FF 00 00 16 4B 01 01 04 00 08 04 64 FD 6B 36    //Response - tag detected!
```

No Card detected:
RM -> PC

```
FF 00 00 16 4B 00       // Response - tag isn't detected!
```

m) I want to use the T = CL APDU Frame. Is this supported ?

Yes, the PN53x reader family supports the ISO14443-4 (T=CL) protocol.
Also called wrapped ISO7816 APDU frame.
All normal processor based cards uses this ISO14443-4 (T=CL) protocol.

The PN531 has a limitation according the loaddata length but this problem is solved in the PN532 (Short APDU frames are supported).

n) What can be the reason if the reader module sends absolute no packet back to the host and this depending on the selected UART baud rate ?

This can happen if you are using a variant equipped with ARYGON Controller and PN531 or PN532. Then if you have selected different baud rates between host to ARYGON controller and ARYGON Controller to TAMA.

Condition:

The TAMA (PN531 or PN532) is running with 115,2 kBaud (PN532 starts always with 115,2 kBaud as default). To be downwards compatible to our existing reader modules with the PN531 our µC provides to the host still 9600Baud. But to the TAMA side we have now always 115,2kBaud as default.
Also the RSTPDN Pin have we to invert in case of PN532.

Mifare cards and other cards commands works, and our controller has no problem
with different baud rate conversion from in normal cases (short TAMA frames). So far OK.

But if we start NFC peer to peer communication with a packet size near to 256 Byte, we receive no ACK packet from the tama and the other side receives always and reproducible the NFC protocol error 00 00 FF 03 FD D5 41 13 D7 00 !

Reason for this:
According both documents (TAMA User Manual for the PN531 and PN532) on chapter "Data Link Level" there you find the following possible reasons for a not send Ack packet from the tama to the host:

LCS Error, DCS error, Framing error or **Timeout in case of UART (HSU)**:

To save time and to avoid having one protocol frame buffer handling more in the system, after our ARYGON controller receives the first byte with 9600 Baud, we send directly this byte with 115,2 kBaud to the TAMA (In case of peer to peer communication we loop through all commands).
Then we wait for the next byte coming with 9600 Baud from the host, then we send again this byte immediately to the TAMA and so on.....

The consequence is, that we need about 20ms more time as calculated and checked by the TAMA in case of 115,2 kBaud. If we exceed the time interval corresponding of four times the duration of a 256 Byte length frame with the current baud rate used then the TAMA sends nothing to the host (to our controller in case of our Reader modules).

That is exactly the case if we have 9600 Baud to the host side and 115,2 kBaud to the TAMA side.

Work around:

That means our module equipped with ARYGON Controller and the PN532 can not exchanged without any additional manipulating of the baud rates !

The surest method will be to switch via the ARYGON high level language the TAMA baud rate to the same value as the ARYGON controller has to the host side.
Then NFC peer to peer will work. This is valid for modules equipped with ARYGON Controller and PN531 or PN532 with UART interface. In case of reader modules with ARYGON Controller with I2C host interface, it should be checked that the host I2C Clock is near to 100 kbit/s (kHz).

o) Why is the wake up handshake part of the order information and when is it necessary to order a module with wake up handshake enabled?

The TAMA wake up handshake mechanism is enabled or disabled by a boot config resistor. This boot config resistor is mounted or not depending on the order information Y or N (refer to the datasheets).

If the wake up handshake is enabled, then a module with TAMA only will not work if the host does not support the additional wake up handshake lines **IRQ I**nterrupt **R**equest and **HReq H**ost **R**equest.
That is the reason why this appears in the order information in module variants **without** ARYGON µC.

With ARYGON µC equipped the Host wake up handshake is always enabled an can be used.
With ARYGON µC equipped the module works also if the host does not support the additional wake up handshake. That means with ARYGON µC equipped the order info Y/N is not relevant because it works in all cases without the need of hardware changes.

Necessity of the wake up handshake:

The sense of the wake up handshake is to wake up the host and the reader module synchronized. If the TAMA wake up source RF Level detector is on, then the TAMA can be waken up if an external RF Field is detected. Normally the TAMA wants then to send immediately the NFC Response packet to the host in this case. But if the host is at sleep, then some bits of the first Response byte will be lost because the host clock is off. To avoid this if wake up handshake is enabled the TAMA warn the host via IRQ line that a response packet is available. The host will awaken if IRQ is connected to a host interrupt Input pin. First after the Host is awaken and does an acknowledge via HReq, the TAMA will send the stored response.

But if wake up handshake is enabled via the boot config resistor, the TAMA will never send the first NFC response to the host without acknowledge via HReq !

Wake up handshake is necessary if both, host and TAMA, wants to go into sleep mode and the TAMA will be awaken by an external RF Field first and wants to wake up the host.

The TAMA sleep command works independent of the wake up handshake mechanism. That means if only the host wants to set the TAMA into and out of the sleep mode, then the wake up handshake mechanism not mandatory.

4. References

PN531 User Manual

Hardware Application Note, *ANSW-ACMA-XX*

Software Application Note, *ANSW-ACMA-APPA-XX*